Docket No.: 1509-458 **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of :
:
Matthew M. WILLIAMSON et al. :
:
U.S. Patent Application No. *Unassigned* :
:
Filed: October 20, 2003 :

For: PROPAGATION OF VIRUSES THROUGH AN INFORMATION TECHNOLOGY NETWORK

CLAIM OF PRIORITY AND TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicant hereby claims, in the present application, the priority of U.K. Patent Application No. 0224396.2, filed October 19, 2002. The certified copy is submitted herewith.

Respectfully submitted,

LOWE HAUPTMAN GILMAN & BERNER, LLP

Allan M. Lowe
Registration No. 19,641

1700 Diagonal Road, Suite 310
Alexandria, Virginia 22314
(703) 684-1111 AML/gmj
Facsimile: (703) 518-5499
Date: October 20, 2003

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

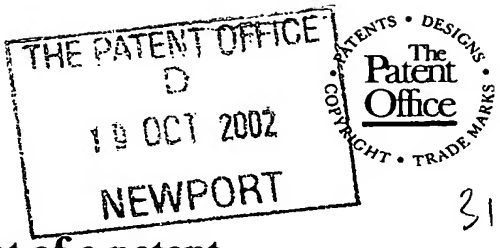
Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed P. Mahoney

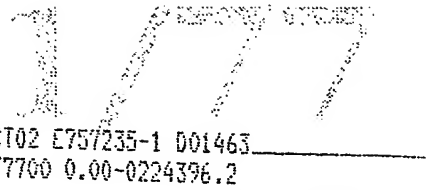
Dated 23 June 2003

Patents Form 1/77

Patents Act 1977
(Rule 16)



31



Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
South Wales
NP10 8QQ

1. Your reference — 200207546-1 GB

2. Patent application number *(The Patent Office will fill in this part)* — 0224396.2 — 19 OCT 2002

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

   Hewlett-Packard Company
   3000 Hanover Street
   Palo Alto
   CA 94304, USA

   Patents ADP number *(if you know it)*

   Delaware, USA

   If the applicant is a corporate body, give the country/state of its incorporation

   4965880004

4. Title of the invention — Propagation of Viruses Through an Information Technology Network

5. Name of your agent *(if you have one)*

   Bruce G R Jones
   Hewlett-Packard Ltd, IP Section
   Filton Road, Stoke Gifford
   Bristol BS34 8QZ

   "Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

   1933005

   Patents ADP number *(if you know it)*

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

   a) any applicant named in part 3 is not an inventor, or
   b) there is an inventor who is not named as an applicant, or
   c) any named applicant is a corporate body.
   See note (d))

   Yes

PROPAGATION OF VIRUSES THROUGH AN INFORMATION TECHNOLOGY NETWORK

The present invention relates to the propagation of viruses through a network of interconnected processing entities.

In current network environments virtually any processing entity (or "host") is at one time or another connected to one or more other hosts. Thus for example in the case of an IT environment, a host in the form of a computer (such as a client, a server, a router, or even a printer for example) is frequently connected to one or more other computers, whether within an intranet of a commercial organisation, or as part of the Internet. Alternatively, in the case of a communications technology environment, a host in the form of a mobile telephone is, merely by virtue of its intrinsic purpose, going to be connected to one or more other hosts from time to time, and an inevitable result is that the opportunities for the propagation of viruses are enhanced as a result. For example in the case of a computer virus known as the "Code Red" virus, once assimilated within a host the virus operates to generate Internet Protocol ("IP") addresses of other potential hosts at random, and then instructs the host to send a copy of the virus to each of these randomly-generated IP addresses. Although not all of the potential hosts are genuine (since the IP addresses are randomly generated), sufficient of the randomly generated addresses are real addresses of further hosts to enable the virus to self propagate rapidly through the Internet, and as a result to cause a substantial drop in performance of many commercial enterprise's computing infrastructure.

Within the context of this specification a virus is data which is assimilable by a host to cause a deleterious effect upon the performance of either: the aforesaid host; one or more other hosts; or a network of which any of the above-mentioned hosts are a part. Thus for example, a virus may act by becoming assimilated within a first host, and subsequent to its assimilation may then cause deleterious effects within that first host, such as corruption and/or deletion of files. In addition the virus may cause self-propagation to one or more further hosts at which it will then cause similar corruption/deletion and further self-propagation. Alternatively the virus may merely be assimilated within the first host and cause no deleterious effects whatsoever, until

communications established by one or more hosts with other hosts within the network is monitored to determine whether the or each monitored host is infected, and upon detection of an infection, data by which communications from the or each monitored host are established with, and possibly additional data besides, are recorded. Depending upon the precise nature of the data recorded, it may then be used to establish *inter alia*: the nature of the virus, the manner in which it propagates.

Embodiments of the invention will now be described, by way of example, and with reference to the accompanying drawings, in which:

Fig. 1 is a schematic representation of one form of network architecture;

Fig. 2 is a schematic illustration of the conventional operational architecture of a computing entity forming a part of, for example, the network of Fig. 1;

Fig. 3 is a schematic illustration of establishment of a connection in accordance with an application protocol from Fig. 2;

Fig. 4 is a schematic illustration of data transmission in accordance with a further application protocol from Fig. 2

Fig. 5 is a schematic illustration of an operational architecture according to an embodiment of the present invention of a computing entity forming a part of a network;

Fig. 6 is a graphical representation of the operation of an embodiment of method according to the present invention;

Fig. 7 is a flowchart illustrating the operation of the method of Figs. 6

Figs. 8A and B are flowcharts illustrating further aspects of embodiments of method according to the present invention.

above, and in conjunction with Internet Protocol (IP). Finally, the network stack 400 includes a system programme known as a driver 410 for the network card, which in essence is low level software that controls the network card.

In the present illustrated examples, the process of establishing a connection in accordance with HTTP will be considered. Usually a request for such a connection is made by the web browser application programme, and this in turn is most likely to be at the behest of a user operating the web browser. Where this is the case, the request will identify the address or "URL" within the network of the computing entity with which a connection is sought, initially using alphanumeric characters entered at the address bar of the browser application programme (for example http://www.hp.com). Ultimately however these are "resolved" into a numerical "IP address" of the form: xxx.xxx.xxx.xxx, where x is either an integer or no number at all and each set of three X's is an integer no greater than 255. An example of an IP address is 192.168.2.2. The IP address is subsequently further resolved into what is known as a physical, or Media Access Control ("MAC") address of the network card of the destination computing entity. Resolution of the URL into an IP address, and the IP address to a MAC address usually takes place at dedicated computing entities within the network, in a manner which is well known *per se*, and will not be described further herein. This description of the connection process in accordance with HTTP, well known *per se*, has described connections legitimately requested by a user, and by means of a URL. However it should be appreciated that it is possible for example to request a connection from the web browser application programme using an IP address, rather than the alphanumeric characters of the URL. This is an aspect of the system behaviour which has been exploited by viruses, some of which randomly generate IP addresses in accordance with the rules governing their allowable format, and then seek connection to those randomly generated addresses.

In the context of the present application it should be appreciated that the term "connection" is a term of art, and is used to refer to a manner of transmitting messages in which acknowledgement of receipt of data is required, so that in the absence of an acknowledgement the connection is deemed either not to have been established, or to have failed, and the transmitted message deemed not to have arrived. On application protocol which operates using connections is HTTP, and an

connections using the same protocol this may be used to distinguish one such connection from the other; a flag indicating that the synchronisation status of the requesting entity is set to "on" (meaning that sequence numbers - which indicate the order of the packet in a total number of packets sent - between the requesting and destination computing entity are to be synchronised), and an initial sequence number 50 (this could be any number). Upon receipt of this packet, the destination machine sends back a packet P20 identifying the source port as 80, the destination port as 3167, a flag indicating that the acknowledgement status is "on", an acknowledgement number 51which augments the sequence number by one, and its own synchronisation flag number 200. When the requesting entity receives this packet it returns a further packet P30 once again identifying the source and destination ports, and a flag indicating that its acknowledgement status is on, with an acknowledgement number 201 (i.e. which augments the sequence number by one). Once this exchange is complete, a connection between the client and server entities is defined as being open, and both the client and server entities send messages up through their respective network stacks to the relevant application programmes indicating that a connection is open between them. In connection with the socket, it should also be noted that the socket comprises an area 460 allocated to store the actual body of the message which it is desired to transmit (sometimes known as the outbound message content, or the outgoing payload), and similarly a further area 470 allocated to store the body of messages which are received (inbound message content, or incoming payload).

When the outgoing payload is to be transmitted, the TCP layer breaks it up into packets (i.e. data structures such as those illustrated above in Fig. 3, but further including at least part of the payload), and the IP layer attaches an IP address header. When an incoming message arrives, it passes up through the network stack, i.e. from the network card 340, up through the Internet Protocol software, etc., and is written in to the relevant socket (as identified, *inter alia* from the port number), from which the application programme retrieves the incoming payload.

Data may alternatively be transmitted using the protocols RSTP/UDP/IP (indicating the hierarchy of protocols in the network stack adopted in conjunction with each other to transmit the data) which do not require a connection; the dispatching entity sends a packet to the destination entity, and does not require an acknowledgement of receipt.

destination hosts. The number of new hosts allowed per time window, and the value of N are determined on the basis of the policy, typically defined by a system administrator, and the policy is preferably formulated to take account of the nature of non virally-related network traffic. In this way, the VPMS operates to monitor the speed at which a virus resident on the host may propagate from that host to other hosts within the network.

Referring to Fig. 6A, over the course of a time window T1, various applications programmes running on the workstation send requests via the VPMS to send data (whether by connection or otherwise) to other hosts within the network ("outbound requests"): the email application programme, which requests dispatch of an email message (having multiple addressees) to a mail server, Mail (Request A) using SMTP, the file management application programme requesting dispatch of a file recording a text document to another user (Request B) via FTP, and the web browser programme which requests connection, (typically via a Web Proxy server), W/Server in order to connect to a site using HTTP (Request C). In the present example, outbound requests to the VPMS from each of these hosts are requests to send data to an identified destination host, and are ultimately manifested by the dispatch of one or more data packets in accordance with the relevant application protocol. The term "request" is intended to be interpreted broadly to encompass any indication (usually from an application programme, although by no means necessarily) that contact with a destination host is sought, and for ease of terminology, the transmission of a request is to be interpreted as indicating that data is transmitted pursuant to a request to transmit such data.

The VPMS operates in accordance with a routine illustrated in Fig. 7, whose features will now be described in more detail in conjunction with Figs. 6A-C, although Fig. 7 should be regarded as a generic illustration of the operation of the VPMS routine, rather than a specific illustration of individual event depicted in Figs. 6. As explained above, the VPMS operates with reference to a series of time intervals, or windows, which in the present example are of constant length. The routine is initiated at step 702 by a clock (typically the clock which defines the time windows) indicating that a time window has commenced. At step 704 the routine then updates a dispatch record, which is a record of the identities of a predetermined number N (which in this

in the dispatch record 612 for this time window T2 is performed in order to establish whether the request is new. The dispatch record however is now a genuine record of the identities of the three hosts contacted most recently during the previous time window T1 (although coincidentally this is identical to the default dispatch record). Upon receipt of Request D, the VPMS routine for that request establishes at step 708 that the identity of this host is not in the dispatch record 612, i.e. that it is a new destination host. It therefore proceeds to step 712, where it adds a copy of the Request D to a virtual buffer whose contents are shown in Fig. 6C, and then ends at 710. In one preferred embodiment, the entire contents of the socket relating to Request D are duplicated in the virtual buffer. However in an alternative embodiment, where for example the payload is large, this is omitted. On receipt of Request B, the VPMS establishes at a step 708 that B is present in the dispatch record, and so the VPMS routine ends at step 710. Request E is also a new request within the time window T2 and so at a step 712 the identity of host E is added to the virtual buffer.

Because receipt of requests are the trigger for the commencement of the routine illustrated in Fig. 7, neither the number of occasions in a given time window in which the VPMS routine is run, nor the timing of their commencement can be known in advance. Additionally, as illustrated in Fig. 7, it is possible for two (or indeed more, although only two are illustrated in Fig. 7) routines to be running in temporal overlap, since one may still be running when another is triggered by a further request. Similarly, a request may trigger the execution of the routine of Fig. 7 just prior to the end of a time window (a situation also illustrated in Fig. 7, with steps which occur at the end of a time window/the beginning of a subsequent time window being shown in dashed lines), so that the execution of the routine may overlap temporally with a part of the next time window. The approach taken by this embodiment of the present invention to this issue of overlap is relatively simple: if at the commencement of time window T_{n+1}, the update of the dispatch record for a previous time window T_n has been completed during the simultaneous running of a VPMS routine commenced in the previous time window T_n, but prior to execution the step 712 (adding a request to the virtual buffer) for that routine, the subsequent update of the virtual buffer in that step 712 will be treated as if performed for a request received in the current time window T_{n+1}. This approach has the benefit of being simple, although it may on

example, following the transmission of the Request D, is one (i.e. the single Request E). Thus, in the same way that the dispatch record is a record of all requests which have been transmitted in accordance with policy, the virtual buffer is effectively a record of all requests which have been transmitted outside that policy.

The role of the virtual buffer is to enable a determination to be made with regard to whether the host upon which the VPMS is running is virally infected. One way in which this can be manifested is the size of the virtual buffer. A state of viral infection may therefore be defined in terms of the size of the buffer, and the stage of any such viral infection by the rate of change of the buffer size. This follows from the generally different behaviour of virally-related and non virally-related network traffic, in that non virally-related or "legitimate" network traffic usually involves contacting only a relatively small number of new destination hosts, whereas, because viruses tend to propagate by transmission to as many disparate destination hosts as possible, an instance of a large number of requests to contact a new destination host will typically be indicative of viral infection. The virtual buffer may be thought of as a queue of virtual new requests waiting for opportunities to be virtually transmitted in accordance with policy (since their "counterpart" real requests are simply transmitted without hindrance). The size of the virtual buffer is therefore one indication of whether there is viral infection, since a large buffer size is indicative of a large number of requests to contact a new host within a short space of time. An alternative indication of viral infection may be the existence of an increasing buffer size. Conversely, generally speaking a buffer size which is steadily declining from a relatively high value may be indicative of a temporary increase in legitimate traffic levels. It can be seen therefore that buffer size may be used to interpret the existence of viral infection with varying levels of complexity, the interpretation typically being something which is defined in the policy.

A second buffer management routine, illustrated in Fig. 8B monitors the virtual buffer, and is triggered by performance of step 814 from the routine of Fig. 8A, i.e. an update in the value of the variable LogNo, following which, at decision step 842, the routine determines whether the size of the buffer is greater than a quantity V_i, which the policy has determined represents viral infection, whereupon at step 844 it generates a virus alert. This may simply be a visual alert to a user of the workstation

substantial number of the requests transmitted during the course of time interval T4 have been virally related.

In the event that a viral warning is generated, various further actions may then be taken, the majority of which are directed toward finding out more about the nature of any possible virus. Specifically the type of information sought may typically include: the destinations to which a virus has been propagated, where applicable the application programme or programmes which it uses to propagate itself, and the action and behaviour of the virus. The nature of the information which may obtained directly from the virtual buffer, or which may be deduced therefrom depends to an extent upon the nature of the data stored in the virtual buffer, and the operating system of the host concerned. For example in the case of one preferred embodiment in which the virtual buffer simply copies the socket, including payload, the destination host will be recorded in the buffer, and possibly, in the case where the virus copies itself to the socket as the outgoing payload, also the virus. Additionally, where the operating system records an identifier in the socket denoting the application programme requesting the socket, and an ability to map this process identifier to the requesting application programme after the socket has been closed (remembering that the virtual buffer contains a copy of the socket, while the actual socket is transient since it is used to implement the request to send data and is then deleted), then the application programme responsible for requesting data transmission can be identified. The use of the data in a socket is only one way in which to collect data relating to possible viral infection, and when using sockets, depending upon the extent of the data collected, the of copying of the sockets is likely to vary. For example, if, as referenced above, the fullest data (including e.g. copies of the payload) is to be retained, further copies of the sockets in the virtual buffer (stored for example in a manner which tags them to the copy of the socket in the virtual buffer) are preferably made over time as the contents of the socket changes over time. However, because two functional elements within the host may cause a change in the data in a socket (e.g. the writing of outgoing data to a socket by an application programme, and removal from the socket of outgoing data by the network stack), maintaining a complete record may nevertheless still be difficult simply from observing the contents of sockets.

the mail server will in fact effectively allow the workstation to contact multiple other hosts within the network (i.e. the specified addressees) all of which may be new, even though, in accordance with the routine described in connection with Fig. 7, the outbound carrier request will only count as a single request which may not even be recognised as new if, as may be likely, the mail server is identified in the current dispatch record. In such a situation therefore, if the VPMS operates simply to record in the virtual buffer those new destination hosts to be contacted per time window on the basis only of those destination hosts which are ostensibly identified in the outbound request, the desired monitoring of viral propagation may be circumvented or reduced, because a single outbound request specifying the mail server does not necessarily represent only a single email subsequently propagating through the network after processing and forwarding by the mail server.

In a modification of the embodiment thus far described therefore, the VPMS includes within its routine a step of identifying the application programme by which an outbound request has been generated. Because certain applications programmes are more likely than others to use outbound carrier requests which invoke the use of a proxy (for example the above-mentioned instance of email, or the case of a web browser programme) it is possible in advance to specify criteria, based on the provenance of an outbound request, identifying those outbound requests likely to be carrier requests. If the packet is generated by one such specified application programme, then the VPMS invokes the use of the application protocol concerned to reveal the identities of the destination hosts specified in the sub-requests; here the eventual addressees for whom the email message is intended. Once the identities of the genuine or ultimate addressees have been obtained, there are several options for processing the request. In accordance with one alternative the identities of the destination hosts specified in the sub-request can be regulated in accordance with the same policy which applies to all other requests, and they can be matched against the host identities within the dispatch record in the manner previously described in the embodiment of Figs.

Since in the case for example of email, the use of outbound carrier requests to a host acting as a proxy for the ultimate addressees of the email messages is the norm, it is, in a modification, possible for different versions of VPMS to run simultaneously,

CLAIMS

1. A method of monitoring propagation of viruses within a network of hosts comprising the steps of:

establishing a record which is at least indicative of identities of hosts within the network to whom data has been sent by a first host ("destination hosts");

during a first time interval, comparing (a) identities of destination hosts identified in requests to send data from the first host and (b) identities of destination hosts identified in the record;

transmitting all requests to send data;

storing in a buffer data relating to requests which identify a destination host not in the record.

2. A method according to claim 1 wherein the record is established by monitoring identities of destination hosts to whom requests have been transmitted during a second time interval, which precedes the first time interval.

3. A method according to claim 2, wherein the record contains a predetermined maximum number of destination host identities, the maximum number being defined in accordance with a policy.

4. A method according to claim 3, wherein the policy additionally defines a maximum number of destination host identities not in the record, to whom requests may be legitimately transmitted in accordance with policy.

5. A method according to claim 4 further comprising the step, at the end of any given time interval, of deleting from the buffer data relating to requests transmitted during the given time interval in accordance with policy.

6. A method according to claim 5 further comprising the step, at the end of the given time interval, of updating the record to reflect identities of hosts identified in requests which are transmitted in accordance with policy during the given time interval.

15. A method according to claim 14 wherein at the end of a time interval, socket data containing identities of destination hosts in respect of whom sockets have legitimately been created is deleted from the buffer.

16. A method according to claim 10 further comprising the step, in the event that the number of socket data items stored in the buffer exceeds a predetermined value, of storing outgoing packets from the first host.

17. A method according to claim 16 wherein packets having a designated destination IP address are stored.

18. A method according to claim 17 further comprising the step of establishing the predetermined IP address from the socket data stored in the buffer.

19. A method according to claim 10 further comprising the step, in the event that the number of socket data items stored in the buffer exceeds a predetermined value, of storing incoming packets to the first host.

20. A method according to claim 19 wherein packets having a designated source IP address are stored.

21. A method according to claim 20, further comprising the step of establishing the predetermined IP address from the socket data stored in the buffer.

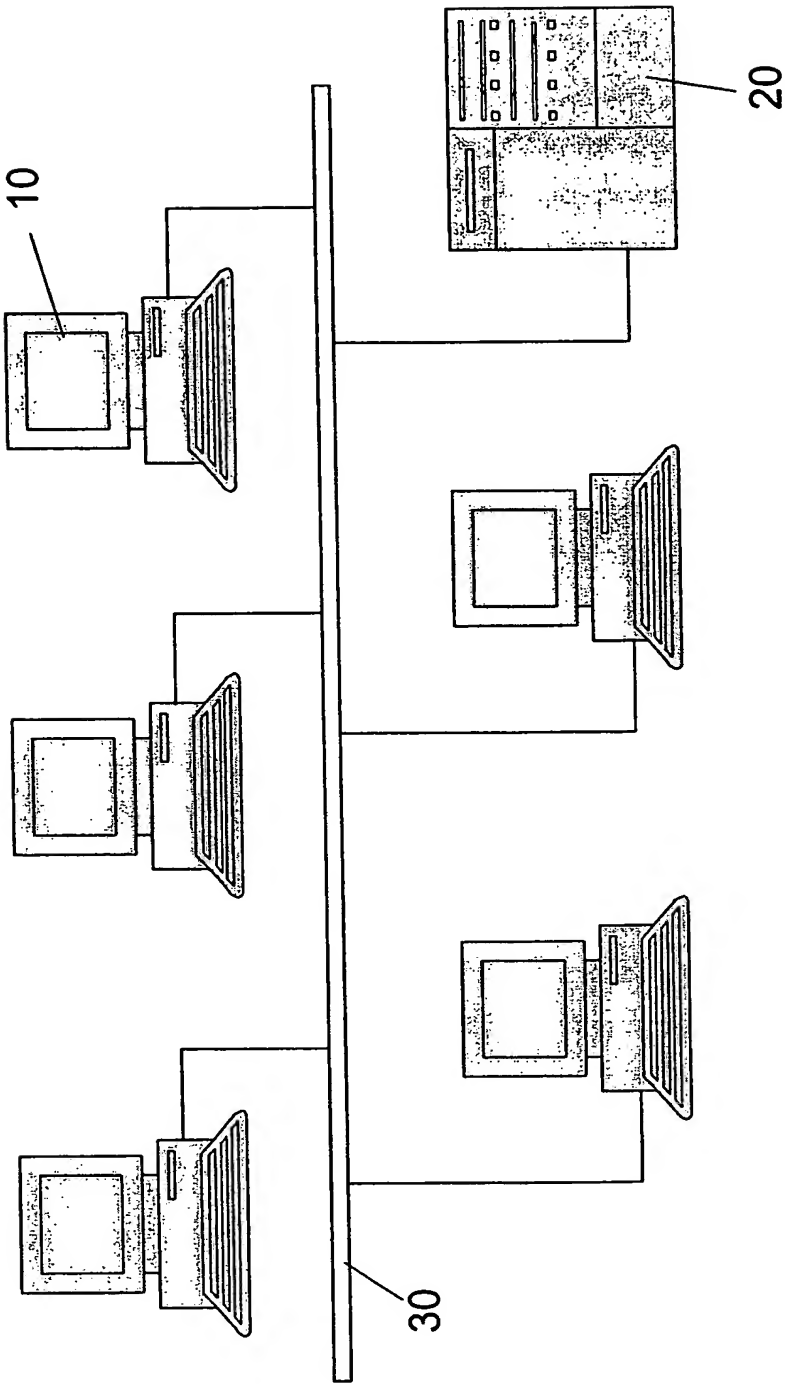


Fig. 1

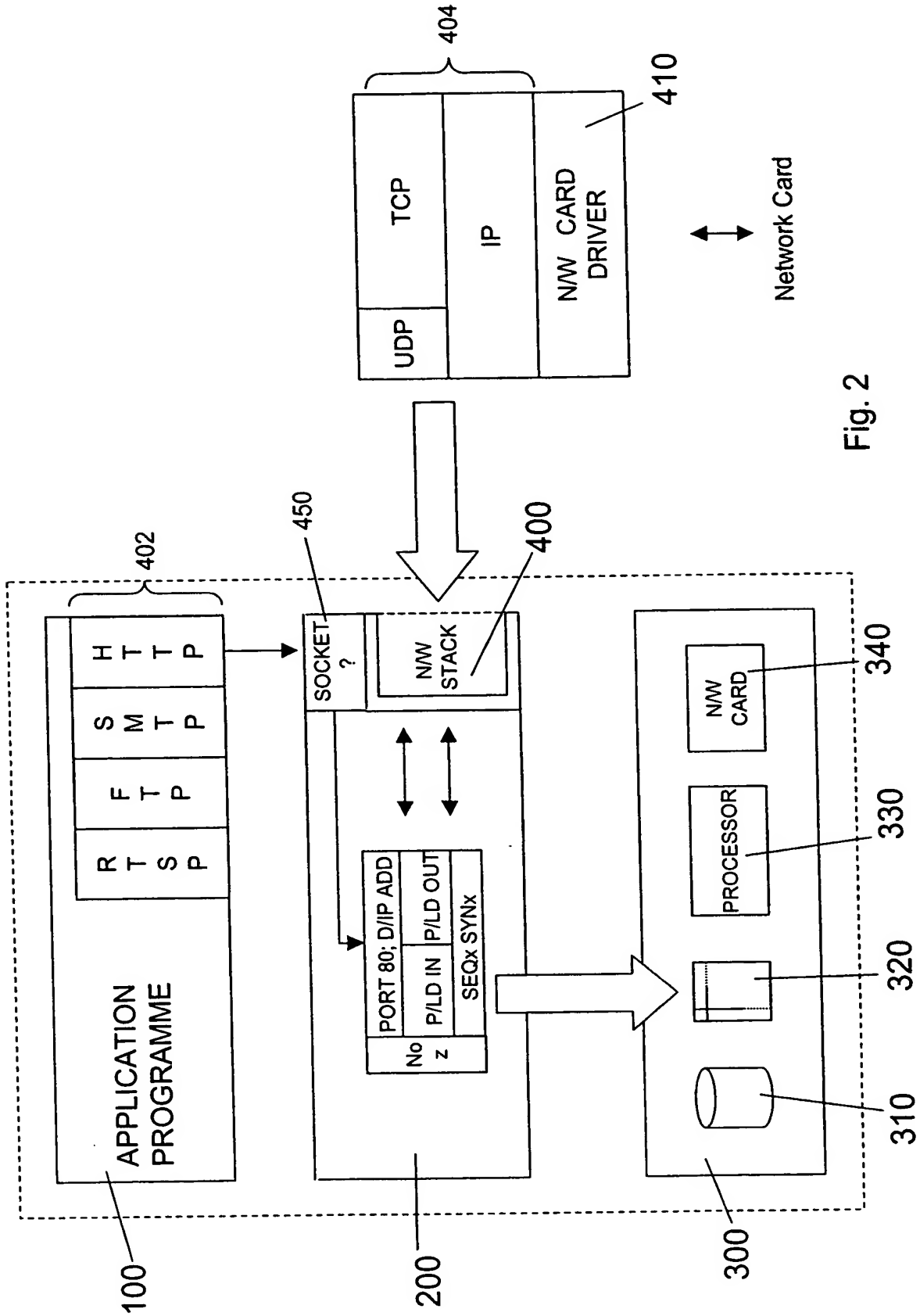

404
410
TCP
IP
N/W CARD DRIVER
UDP
Network Card
450
400
402
SOCKET ?
N/W STACK
HTTP
SMTP
FTP
RTSP
APPLICATION PROGRAMME
PORT 80; D/IP ADD
P/LD IN
P/LD OUT
SEQx SYNx
No z
N/W CARD
PROCESSOR
340
330
320
310
100
200
300

Fig. 2

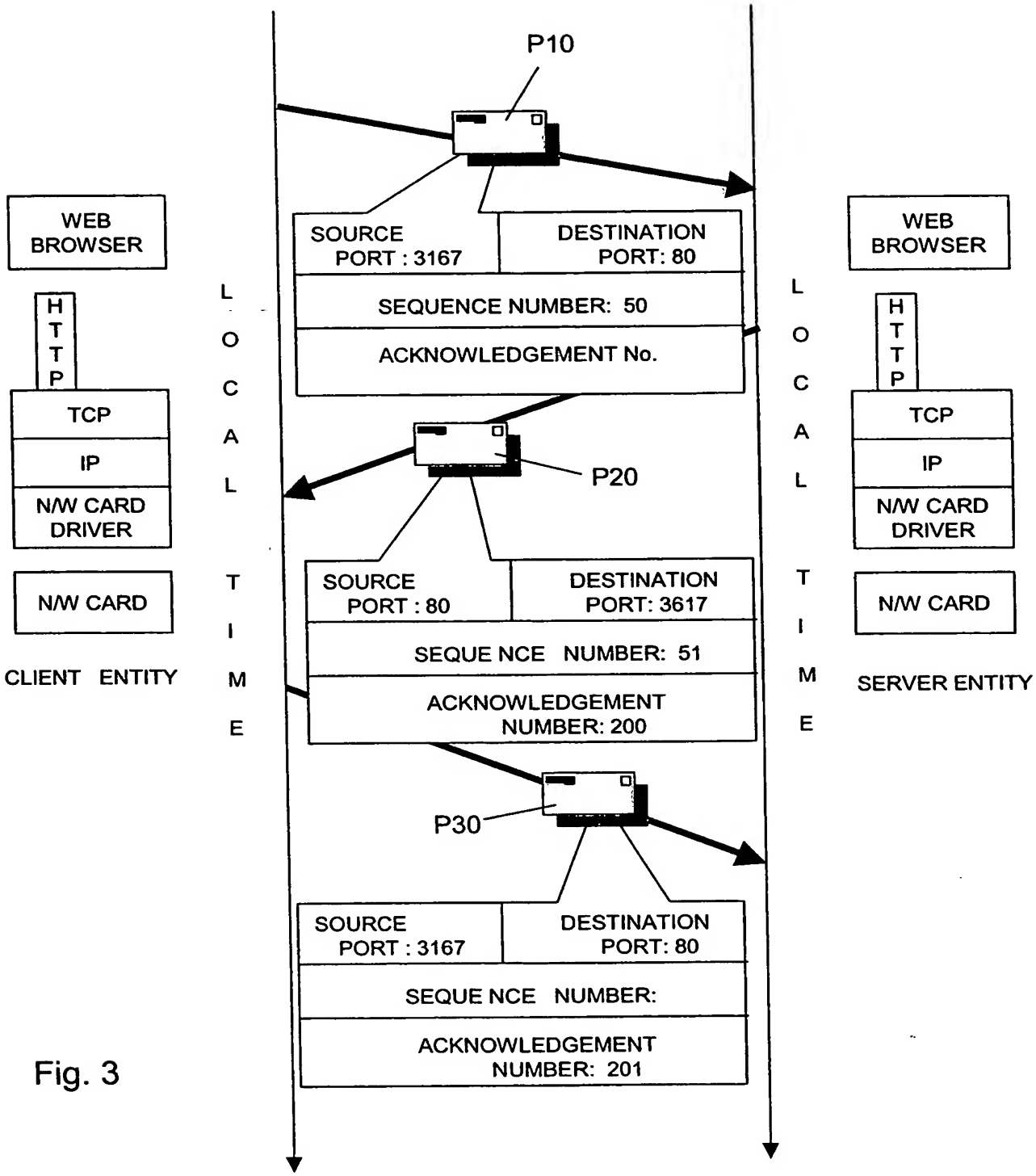

P10
WEB BROWSER
SOURCE PORT : 3167
DESTINATION PORT: 80
SEQUENCE NUMBER: 50
ACKNOWLEDGEMENT No.
WEB BROWSER
H T T P
TCP
IP
N/W CARD DRIVER
N/W CARD
L O C A L T I M E
P20
SOURCE PORT : 80
DESTINATION PORT: 3617
SEQUE NCE NUMBER: 51
ACKNOWLEDGEMENT NUMBER: 200
P30
SOURCE PORT : 3167
DESTINATION PORT: 80
SEQUE NCE NUMBER:
ACKNOWLEDGEMENT NUMBER: 201
CLIENT ENTITY
SERVER ENTITY

Fig. 3

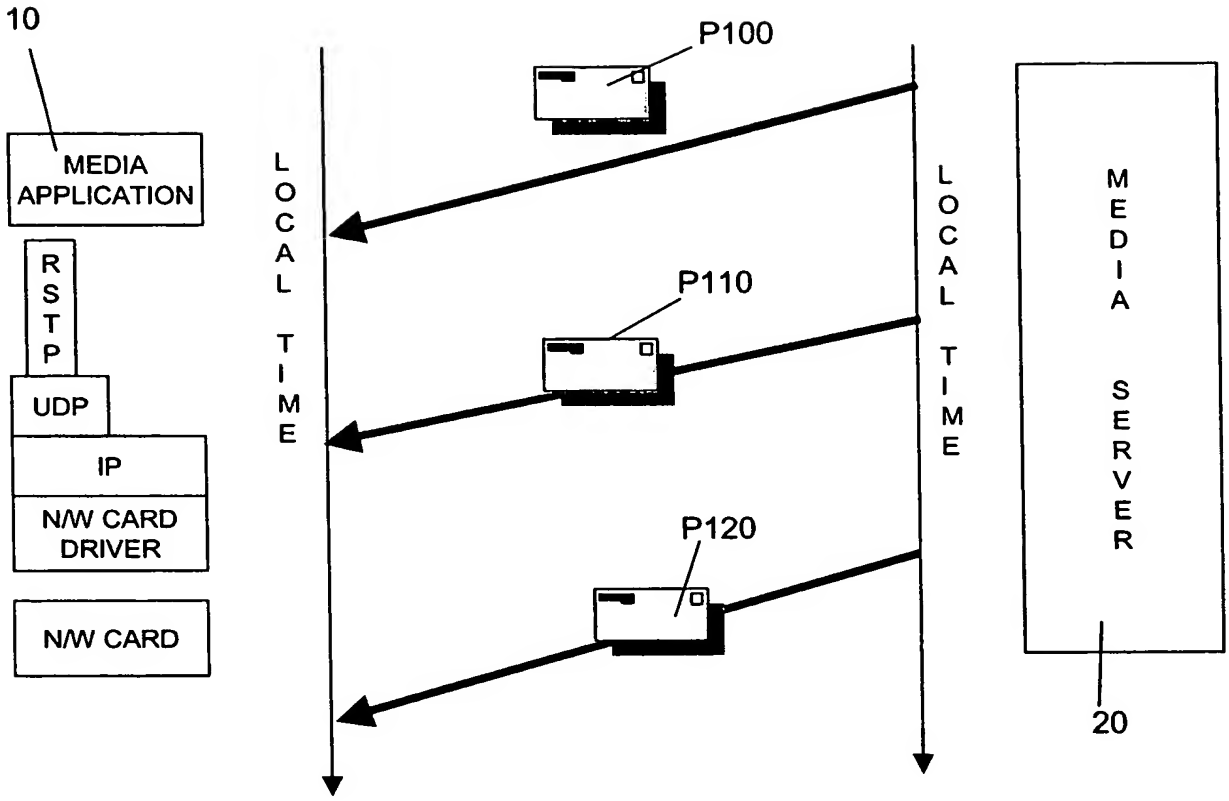

10
MEDIA APPLICATION
RSTP
UDP
IP
N/W CARD DRIVER
N/W CARD
LOCAL TIME
LOCAL TIME
P100
P110
P120
MEDIA SERVER
20

Fig. 4

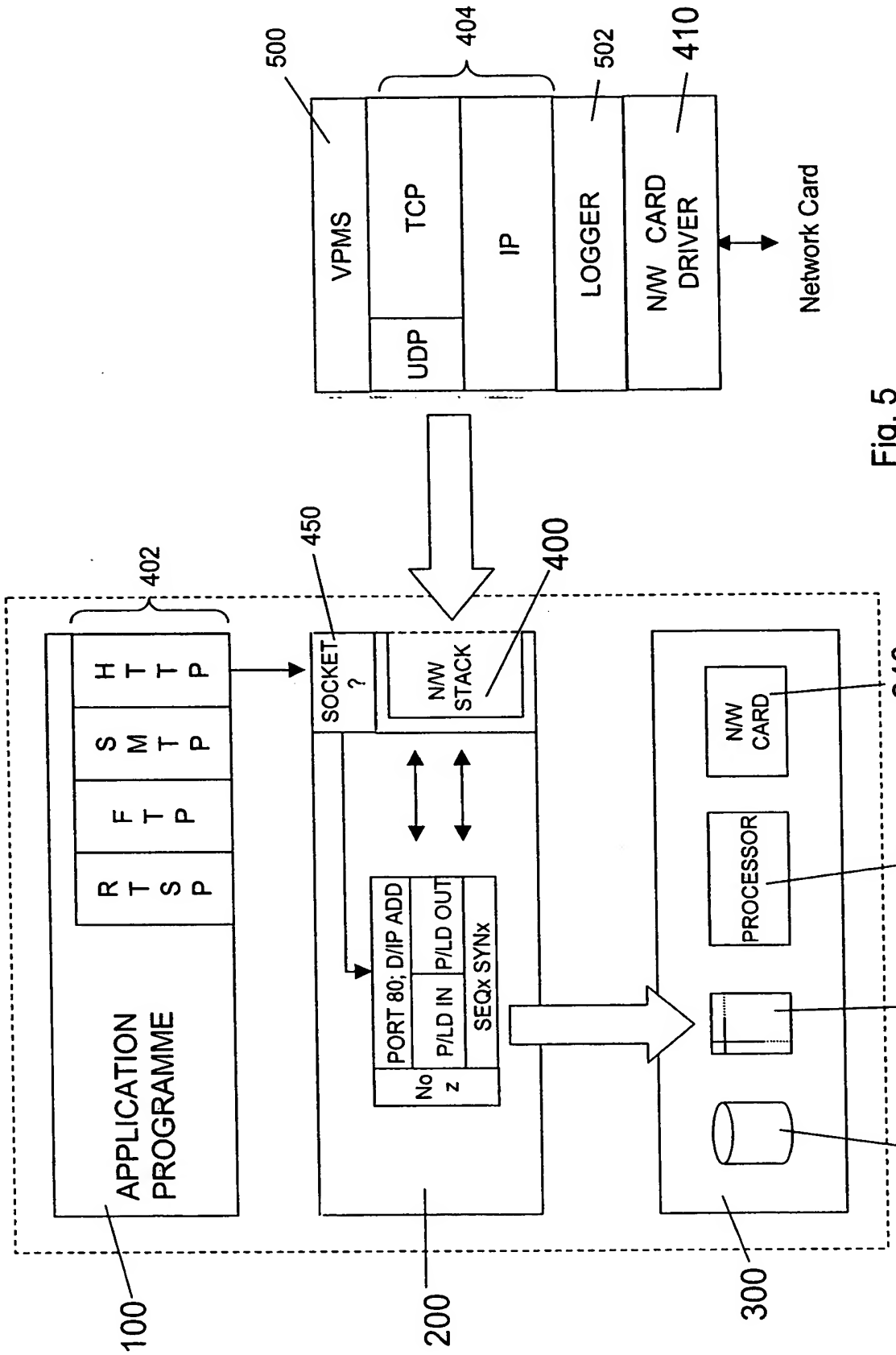
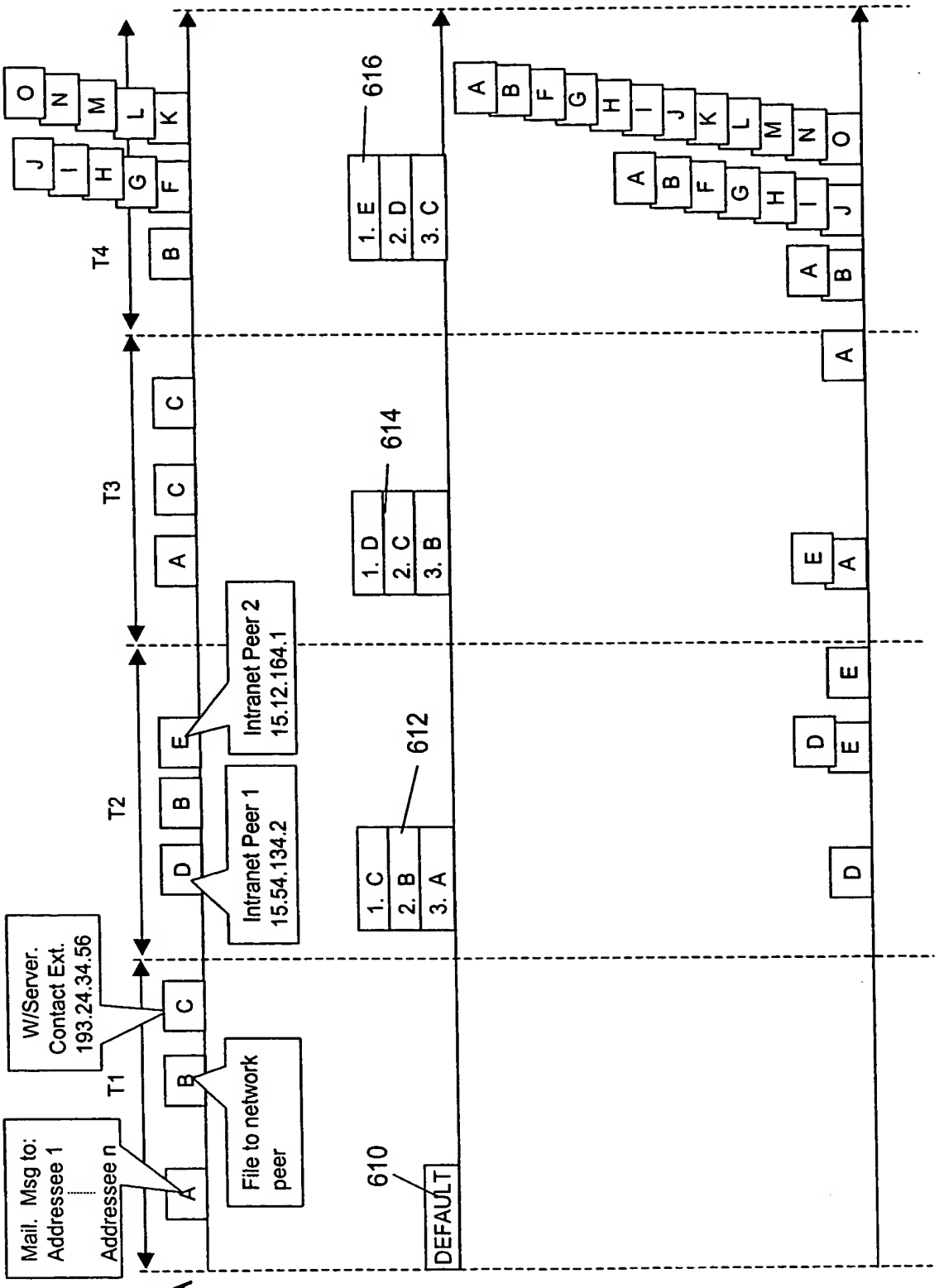

500
404
502
410
VPMS
TCP
UDP
IP
LOGGER
N/W CARD DRIVER
Network Card
402
450
400
APPLICATION PROGRAMME
HTTP
SMTP
FTP
RTSP
SOCKET ?
N/W STACK
PORT 80; D/IP ADD
P/LD IN
P/LD OUT
SEQx SYNx
No z
N/W CARD
PROCESSOR
340
330
320
310
100
200
300

Fig. 5


Mail. Msg to:
Addressee 1
Addressee n
W/Server.
Contact Ext.
193.24.34.56
File to network
peer
Intranet Peer 1
15.54.134.2
Intranet Peer 2
15.12.164.1
DEFAULT
610
612
614
616
T1
T2
T3
T4
1. C
2. B
3. A
1. D
2. C
3. B
1. E
2. D
3. C
Fig. 6A
6B
6C

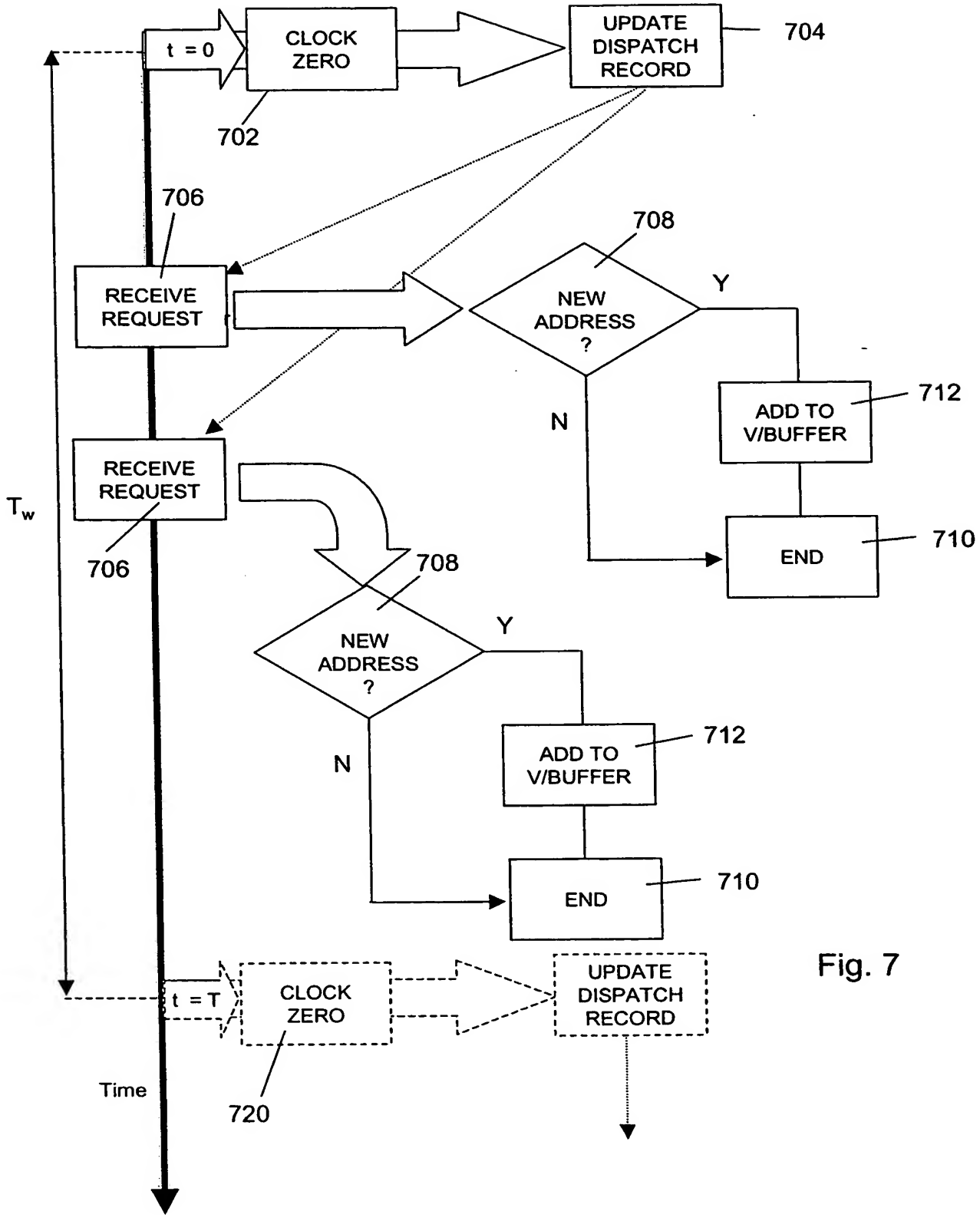

t = 0
CLOCK ZERO
UPDATE DISPATCH RECORD
704
702
706
708
RECEIVE REQUEST
NEW ADDRESS ?
Y
N
ADD TO V/BUFFER
712
RECEIVE REQUEST
706
END
710
T_w
708
NEW ADDRESS ?
Y
N
ADD TO V/BUFFER
712
END
710
t = T
CLOCK ZERO
UPDATE DISPATCH RECORD
Time
720

Fig. 7

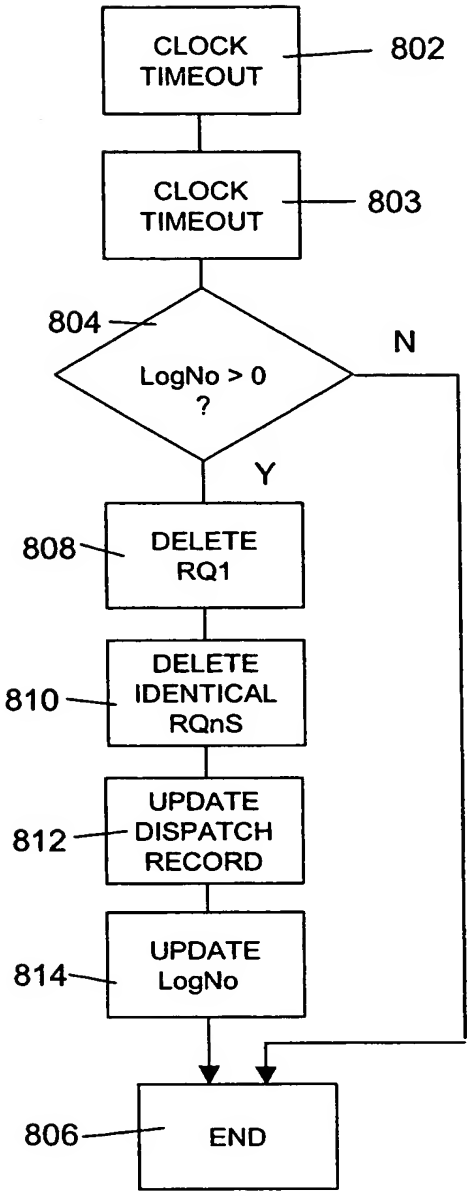

CLOCK TIMEOUT
802
CLOCK TIMEOUT
803
804
LogNo > 0 ?
N
Y
808
DELETE RQ1
DELETE IDENTICAL RQnS
810
UPDATE DISPATCH RECORD
812
UPDATE LogNo
814
806
END

Fig. 8A

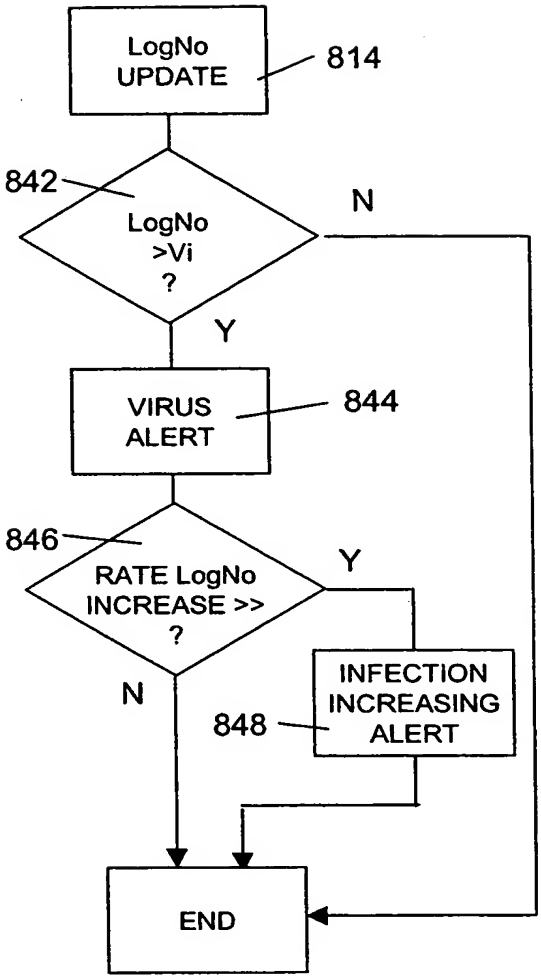

LogNo UPDATE
814
842
LogNo >Vi ?
N
Y
VIRUS ALERT
844
846
RATE LogNo INCREASE >> ?
Y
N
INFECTION INCREASING ALERT
848
END

Fig. 8B